Bakalářská práce

VŠB – TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
KATEDRA KYBERNETIKY A BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

Aplikace pro zobrazování biologických signálů v mobilních technologiích

Mobile Application - Biological Signal Display

2017 Kamila Smičková

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Kamila Smičková**

Studijní program: B2649 Elektrotechnika

Studijní obor: 3901R039 Biomedicínský technik

Téma: Aplikace pro zobrazování biologických signálů v mobilních technologiích
Mobile Application - Biological Signal Display

Jazyk vypracování: čeština

Zásady pro vypracování:

Tématem bakalářské práce je návrh a implementace aplikace pro zobrazování biologických signálů (např. EKG, EEG, EMG) v mobilním telefonu. Cílem je grafické zobrazování naměřeného vzorku dat - signálu. Podle zvoleného biosignálu může být navíc implementován algoritmus pro výpočet některého z doplňkových parametrů - např. frekvence srdeční činnosti pro EKG.

Body zadání:
1. Analýza dostupných řešení a způsobů realizace mobilních komponent/aplikací pro zobrazování signálů.
2. Návrh aplikace (funkcí, grafického rozhraní a způsobu implementace).
3. Implementace mobilní aplikace.
4. Testování a měření.
5. Zhodnocení výsledků práce a testování.

Seznam doporučené odborné literatury:

[1] CASTLEDINE, Earle, Jakub MUŽÍK, Myles EFTOS a Max WHEELER. *Vytváříme mobilní web a aplikace pro chytré telefony a tablety.* 1. vyd. Brno: Computer Press, 2013, 288 s. ISBN 978-80-251-3763-5.
[2] RYCHLÝ, Marek a Jaroslav ZENDULKA. *Modelling of Component-based Systems With Mobile Architecture: Monograph.* Vyd. 1. Brno: Faculty of Information Technology, Brno University of Technology, 2010, xii, 139 s. ISBN 978-8021442115.
[3] GOVE, Darryl a Lukáš KREJČÍ. *Programování aplikací pro vícejádrové procesory.* Vyd. 1. Brno: Computer Press, 2011, 416 s. ISBN 978-80-251-3487-0.
[4] LACKO, Ľuboslav a Martin HERODEK. *ývoj aplikací pro Android.* 1. vyd. Brno: Computer Press, 2015, 472 s. ISBN 978-80-251-4347-6.
[5] VLACH, Karel a Jakub JIRKA. *Základy komponentních technologií pro řízení.* Vyd. 1. Ostrava: VŠB - TU Ostrava, 2013. 301 s.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Karel Vlach**

Datum zadání: 01.09.2016

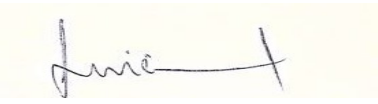
Datum odevzdání: 28.04.2017

doc. Ing. Jiří Koziorek, Ph.D.	prof. RNDr. Václav Snášel, CSc.
vedoucí katedry	*děkan fakulty*

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

„Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.“

V Ostravě dne 28.4 2017



……

Tímto bych chtěla velice poděkovat vedoucímu mé bakalářské práce, panu Ing. Karlu Vlachovi, Ph.D.za veškeré rady a pomoc.

Abstrakt

Obsahem této bakalářské práce je vývoj mobilní aplikace, která může zobrazit naměřený EKG signál v mobilním telefonu. Umožňuje přenositelnost, obecnou dostupnost, výpočet tepové frekvence. Aplikace je vytvořena ve Visual Studiu. Aplikační logika je napsaná v C# a jazyk pro definování uživatelského rozhraní je použitý XAML. Aplikace je pomoci Xamarinu vytvořena pro všechny tři nejrozšířenější platformy-iOS, Android i Windows. Aplikace je následně ověřena na reálném zařízeni Sony.

Cílem bakalářské práce je umožnění vykreslení EKG křivky pacienta i mimo jeho dosah pomocí mobilního telefonu. Teoretická část je zaměřena na analýzu dostupných řešení, jak lze aplikaci vytvořit pro více platforem najednou. Praktická část je zaměřena na vytvoření aplikace pro zobrazování EKG signálu a implementaci algoritmu pro výpočet tepové frekvence.

Klíčová slova

mobilní aplikace, operační systém, Android, Visual Studio, Xamarin, C#, EKG, tepová frekvence

Abstract

The content of this bachelor thesis is the development of a mobile application which can display the measured ECG signal in a mobile phone. Allows portability, general availability, heart rate calculation. The application is created in Visual Studio. The application logic is written in C # and the user interface definition language is used by XAML. It is coded in Xamarin which help application to run for all three most popular platforms - iOS, Android and Windows. The application is then verified on a Sony device.

The aim of the bachelor thesis is to enable the patient's EKG curve to be plotted outside of the patient's range using a mobile phone. The theoretical part is focused on the analysis of the available solutions how to create an application for multiple platforms at one time. Practical part is focused on creation of application for display of ECG signal and implementation of algorithm for calculation of pulse rate.

Keywords

mobile application, operating system, Android, Visual Studio, Xamarin, C#, ECG, BPM

Obsah

Seznam ilustrací

Seznam tabulek

1. Úvod

Moderní mobilní technologie se posouvají dopředu a pronikají do mnoha odvětví. Jedním z nich je i zdravotnictví, kde vzhledem k lepší dostupnosti a přenositelnosti může zásadně ovlivnit nebo zachránit život pacienta. Vykreslení EKG na mobilním zařízení může umožnit rychlejší přistup k naměřeným datům a ke kontrole pacienta i pro laickou veřejnost.

Teoretická část by měla objasnit důvod vytvoření této mobilní aplikace v daném vývojovém prostředí za použití příslušného programovacího jazyka. První část je zaměřena na mobilní aplikace, kde je vysvětlen rozdíl mezi nativní a multiplatformní aplikací. V tomto případě bylo zapotřebí vytvořit multiplatformní aplikaci, aby se dala použit na třech dnes nejrozšířenějších platformách (iOS, Android i Windows). Dále jsou v teoretické části podrobněji popsány operační systémy, vývojová prostředí a programovací jazyky. Další problematikou bylo najít vhodnou knihovnu na vykreslení EKG signálu. Danou aplikaci bylo potřeba vytvořit, aby vykreslila EKG signál, který byl naměřen pomocí vývojového kitu ADS1298 vedoucím bakalářské práce. Další část se věnuje samotnému EKG, jeho vzniku, měření a využití v praxi.

Další kapitoly jsou věnovány praktické části. Nejdříve je popsána samotná mobilní aplikace, co umí, jak se ovládá. Poté je prezentován vývoj mobilní aplikace pro vykreslení EKG a algoritmus pro výpočet tepové frekvence. Algoritmus, který vyvinuli pánové Pan a Tompkins (1985) na přesnou detekci QRS komplexu, se skládá z více filtrů, které jsou zapojeny do kaskády. Filtry jsou v praktické části popsány jak matematicky, tak i pomocí zdrojového kódu. V závěru této bakalářské práce bude aplikace spuštěna na reálném zařízení a výsledky budou zaznamenány.

2. Mobilní aplikace

2.1. Nativní aplikace

Naprogramování nativní mobilní aplikace pro Apple iOS znamená použití programovacího jazyku Objective-C nebo Swift, které slouží pro programování pouze pro iOS. Tahle aplikace by se musela dělat ve vývojovém prostředí Xcode. Stejně tak, kdyby měla být aplikace pro Android, tak by se musela použít technologie Java. Třetí možností by byla aplikace pro Widows Phone v jazyce C# nebo XAML. Z uživatelského hlediska není rozdíl mezi nativní a cross-platformní aplikací. [14]

2.2. Mezi-platformní mobilní aplikace

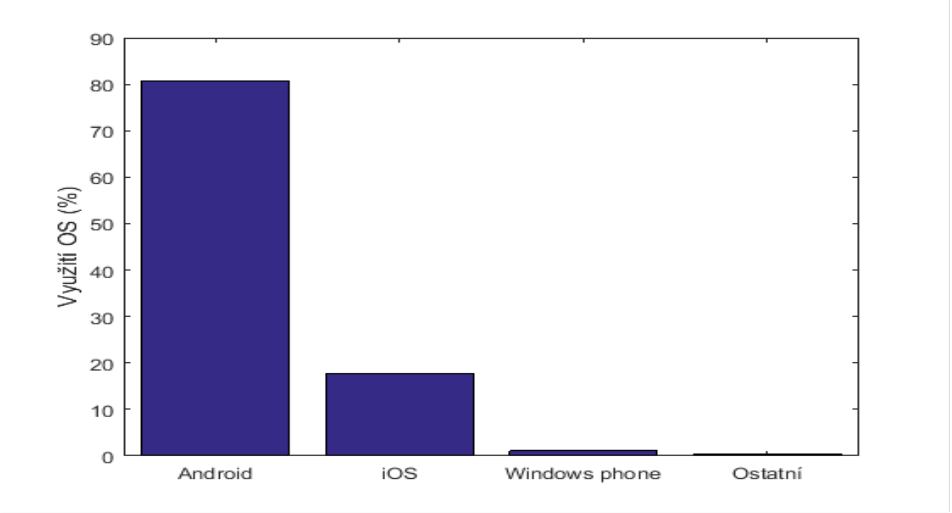
Je důležité pochopit, proč se aplikace nazývá cross-platformní. Faktem je, že samo-spustitelný soubor sestaven pro jednu mobilní platformu nemůže být spouštěn v druhém operačním systému. Cross-platformní aplikace jsou psány v jednom jazyce a v jednom vývojovém prostředí, poté se zkompilují na nativní aplikace. [14]

2.3. Hybridní aplikace

Nejčastěji se jedná o aplikace vyvinuté pomocí webových technologií. Výsledkem je webová aplikace, která běží v "obalu" a neslouží jako webová stránka, ale jako samostatná aplikace. Nevyžaduje žádnou instalaci a má samostatnou ikonu. Hybridní řešení je velmi populární mezi vývojáři multiplatformních aplikací. Je to způsobeno tím, že téměř každý mobilní operační systém může rovněž zpracovávat funkci webového prohlížeče, což znamená, že v případě, že aplikace je již spuštěna v nějakém mobilním OS, pak nebudou žádné potíže jej spustit na jiném OS. [14]

3. Mobilní operační systémy

Mobilní operační systém (OS) je software, který umožňuje smartphonům, tabletům a jiným zařízením spouštění aplikací a programů. Existuje několik OS a pro každý je specifické vývojové prostředí a programovací jazyk. V dnešní době patří k nejrozšířenějším OS Android. Mezi další OS patří iOS , Windows phone, Blackberry a další. V grafu viz Obr. 1 je procentuálně zaznamenáno využití operačních systému.



Obr. 1: Využití operačních systémů

3.1. iOS

Jedná se o operační systém od firmy Apple, který běží pouze na výrobcích od této firmy. V roce 2007 byl představen původně jako iPhone OS, ale od čtvrté verze se používá pojmenování iOS. Pro vytvoření aplikací na iOS je potřeba mít nainstalovanou nejnovější verzi iPhone SDK, která zahrnuje Xcode IDE, iphone simulátor a sadu dalších nástrojů pro vývoj aplikací pro iPhone a iPod touch, ale jde nainstalovat pouze na operačním systému OS X. Tyto nástroje vám pomohou rozvíjet své aplikace a umožní vám je spustit v simulátoru. Aplikace na iPhone se píší především v jazyce Objective-C. [13] [25]

3.2. Android

Android je operační systém, který je postaven na linuxovém jádře. Je to open-source projekt od Open Handset Alliance(OHA). První Android telefon, T-mobile G1, který se prodával také jako HTC Dream, byl vydán v říjnu 2008. Operační systém android má bohatou sadu funkcí. Podporuje 2D a 3D grafiku založenou na specifikaci OpenGL ES 2.0. Je zde dobrá mediální podpora pro běžné audio, pro video a pro obrazové formáty. Operační systém Android podporuje multi-touch vstup, který ale nemusí být podporován na všech zařízeních. Webový prohlížeč je založen na výkonném WEBKIT motoru a zahrnuje V8 JavaScript runtime Chrome. Multitasking je aplikací podporován. U androidu je multitasking řízen strukturováním aplikací jako „činnosti“. Činnosti mají zřetelný vizuální prezentaci a měla by být jednoúčelová, jako je pořizování fotografie, vyhledávání a prezentace výsledků nebo úprava kontaktů. Aktivity by měly být přístupné i z jiných aplikací. Jednoduchá aplikace může implementovat jednu aktivitu, ale složitější aplikace mohou být implementovány jako celá řada aktivit a prezentovat se jako jedna aplikace. Aplikace pro android jsou psané především v jazyce Java, ty ale na platformě

neobsahují Java Virtual Machine, ale Java třídy jsou rekompilovány do bytekódu Dalvik a běží na virtuálním stroji Dalvik. Dalvik byl speciálně navržen pro Android ke snížení spotřeby energie a také, aby se lépe pracovalo s omezenou pamětí. Od vydání Android NDK (Native Development Kit) v červnu 2009, mohou vývojáři také vytvářet nativní knihovny v C a C++ k opětovnému použití stávajícího kódu nebo ke zvýšení výkonu. Nejčastěji používaným a doporučeným vývojovým prostředím je Eclipse s Android Development Tools plug-in. Plug-in zajišťuje plnohodnotné vývojové prostředí, které je integrováno s emulátorem. [13] [11] [7] [23] [25]

3.3. Windows phone

Operační systém Windows Mobile od firmy Microsoft je, oproti ostatním operačním systémům, nejvíce srovnatelný se stolním počítačem. K vytvoření nativní aplikace je nutné mít Microsoft Visual Studio 2008 Professional, Windows Mobile SDK, Windows Mobil 6 Professional, Windows Mobile 6.5 Developer Tool Kit a ActiveSync. Aplikace na Windows Phone se píší v jazyce C #. [13] [9]

Tabulka 1. Výhody a nevýhody OS

	Výhody	**Nevýhody**
iOS	• Odladěnost a rychlost systému • Kvalitní aplikace • Přednostně vydávané aplikace a hry oproti konkurenčním OS • Nulová fragmentace	• Uzavřenost systému, žádné přizpůsobení • Nelze sdílet data s jinými OS
Android	• Otevřenost systému • Velký výběr přístrojů s tímto OS • Kvalitní služby od Googlu • Hodně aplikací zdarma • Multitasking	• Velká míra pirátství díky otevřenosti systému • Velká HW náročnost • Nejméně stabilní systém • Fragmentace systému
Windows Phone	• Plynulý systém • Jednoduché sdílení a přenášení dat • Zabezpečení a stabilita systému	• Malý počet aplikací ve Windows storu

4. Vývojová prostředí

4.1. Xcode

Xcode je integrované vývojové prostředí od firmy Apple určené pro tvorbu aplikací pro Mac OS X a iPhone. Preferovaný jazyk pro tvorbu v Xcode je Objective-C, který je potřebný pro iPhone aplikace. Xcode také podporuje další jazyky, jako je C, C++, Fortran, Java, Objective-C++, AppleScript, Python a Ruby. [4] [25]

4.2. Eclipse

Eclipse je multiplatformní vývojové prostředí, které je určené především pro jazyk Java. Oproti jiným vývojovým prostředím může Eclipse za pomocí pluginů rozšířit seznam programovacích jazyků. V základní verzi obsahuje Eclipse pouze integrované složky pro vývoj standardní Javy, čímž je kompilátor, debugger a další. Ovšem už neobsahuje třeba nástroj pro vizuální návrh grafických uživatelských rozhraní nebo aplikační server, tohle a ostatní rozšíření je zapotřebí dodat pomocí pluginů. Eclipse je momentálně nejrozšířenější vývojové prostředí pro Javu. [3] [23]

4.3. NetBeans

NetBeans je oficiální, multiplatformní vývojové prostředí pro Javu8 od firmy Oracle Corporation. Prostředí je celé naprogramované v jazyce Java, ale je přístupné i pro jiné operační systémy (Microsoft Windows, Linux Mac OS). [3]

4.4. Visual Studio

Visual studio je plnohodnotné integrované vývojové prostředí pro Android, iOS i Windows od firmy Microsoft. Mezi vestavěné jazyky patří C/C++, VB.NET a C#. Visual Studio má editor kódu, který podporuje IntelliSense a redaktorování. Integrovaný debugger tak pracuje jak na úrovni stroje, tak na úrovní kódu. Dále ma Visual Studio vestavěné nástroje, které zahrnují designer formulářů pro tvorbu aplikací s GUI, designer webu, tříd a databázových schémat. [4] [3]

4.5. Lazarus

Lazarus je profesionální open source multiplatformní vývojové prostředí pro rychlý vývoj aplikací. Obsahuje řadu komponent připravených k použití tak, aby mohl uživatel snadno vytvářet složité grafické rozhraní. S Lazarem lze vytvořit prohlížeč souborů, prohlížeč obrázků, databázové aplikace pro

úpravu grafického softwaru, hry, 3D software, software lékařské analýzy a jakýkoliv jiný typ softwaru. Aktuálně běží na systémech Linux, FreeBSD a Win32. Obsahuje přizpůsobitelný editor zdrojového kódu, vizuální tvorbu formulářů s manažerem balíčků, debuggerem a úplnou integrací překladače FreePascal do GUI. [3]

4.6. Delphi

Delphi je integrované grafické vývojové prostředí od firmy Borland, které se specializuje na vytváření aplikací na platformě MS Windows v jazyce Object Pascal, což je objektoví nástavba Pascal. Delphi obsahuje systém RAD, díky kterému je možný vizuální návrh grafického uživatelského rozhraní, na kterém je vytvářena kostra zdrojového kódu. To v podstatě vede k urychlení vývojového cyklu. [3] [4]

5. Programovací jazyky

5.1. Objective-C

Objective-C je primární programovací jazyk, který se používá při vytváření softwaru pro OSX a iOS. To je podmnožinou programovacího jazyka C a poskytuje objektově orientované schopnosti. Objective-C dědí syntaxi, primitivní typy a řízení toku C a přidává syntaxi pro definování tříd a metod. To také přidává podporu jazyka na úrovni pro podporování objektů grafů. Při vytváření aplikací pro OS X a iOS zabere většinu času práce s objekty. [11] [3]

5.2. C#

C# je objektově orientovaný vysokoúrovňový programovací jazyk, který byl vyvinutý firmou Microsoft společně s platformou .NET Framework, dále byl později schválený standardizačními komisemi ECMA (ECMA-334) a ISO (ISO/IEC 23270). Abychom mohli psát program budeme muset nainstalovat prostředí Microsoft .NET Framework 1.1, což je nezávislá platforma, rozhraní mezi OS a programovacím jazykem.NET Framework je prostředí, které spojuje spoustu jazyků, jako C# a VB.NET. Tohle prostředí poskytuje sadu společných knihoven pro všechny jazyky založené na .Net, proto není problém přecházet z jednoho jazyka na druhý. [7] [3]

5.3. Java

Java je objektově orientovaný programovací jazyk. Jde o jeden z nejpoužívanějších programovacích jazyků na světě. Java je programovací jazyk nezávislý na platformě. Vyvíjí ho společnost SUN,

Microsystems a je zdarma dostupný pro různé operační systémy (Windows, Linux, Solaris). Nezávislost na operačním systému a na hardwaru počítače zajišťuje způsob kompilace. Zdrojové kódy programu nejsou překládány do strojového kódu procesoru, ale pouze předzpracovávány do tzv. byte-kódu. Ten ještě není závislý na konkrétním procesoru, ale časově náročné fáze kompilace jsou již provedeny. Takto předzpracovaný kód je pro člověka nečitelný. Při spuštění Java programu je byte-kód velmi rychle převeden na strojový kód daného procesoru (s ohledem na použitý operační systém) – to provádí tzv. Java Virtual Machine (JVM). Odměnou za kompilaci je řádově vyšší výkonnost (rychlost programu). Jazyk Java je podobný jazyku C++. Základní rozdíl je v tom, že Java je plně objektová, neexistují v ní globální proměnné, vše musí být součástí nějakého objektu (nebo třídy). V jazyce Java jsme odstíněni od hardwaru, takže např. nemáme přímou kontrolu nad pamětí a není potřeba uvolňovat alokovanou paměť. Zdrojový kód Java programů můžete psát v libovolném textovém editoru, existují však vývojová prostředí umožňující rychlejší a pohodlnější vývoj aplikace, která jsou v základní verzi zpravidla dostupná zdarma. Jedním z takových je Inprise JBuilder 6. [1] [11] [23]

Java se dělí do tří balíčků:

- Java ME – pro vývoj pro mobilní zařízení
- Java SE – pro vývoj desktopových aplikací
- Java EE – určená pro enterprise aplikace

Tabulka 2: Vývojová prostředí a program. jazyky

	Vývoj. Prostředí	**Programovací jazyk**
iOS	Xcode	Objective-C
Android	Eclipse/Java/Android Development Tool (ADT)	Java
Windows Phone	Visual studio	C #, .NET, Silverlight nebo WPF

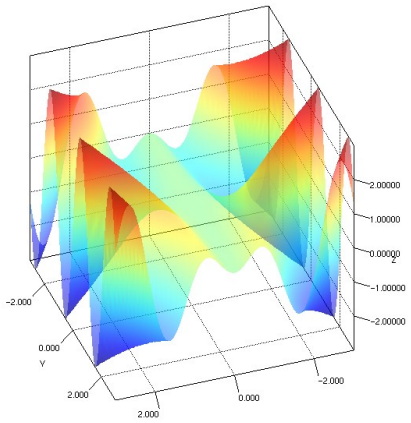
6. Knihovny

6.1. Knihovny pro Javu

Pro programovací jazyk java existuje mnoho knihoven, které jsou zaměřené na grafy. Některé jsou volně dostupné a na některé je nutné zakoupit licenci. V této podkapitole je seznam javovských knihoven.

6.1.1. Jzy3D

Jzy3D je open source 3D knihovna, která vytváří grafy z vědeckých údajů. Je především k bodovým grafům, sloupcovým grafům a dalším primitivním 3D grafům viz Obr. 2. API poskytuje podporu pro bohaté interaktivní grafy s colorbarsy, bublinami a překryvy. Axis a rozvržení grafu lze přizpůsobit a rozšířit. [15]



Obr. 2: Ukázka Jzy3D

6.1.2. Charts4j

Charts4j je zdarma, umožnuje vývojářům programově vytvářet grafy, z hodnot, které jsou v tabulce Tools Google prostřednictvím jednoduchého a intuitivního rozhraní Java API. [16]

6.1.3. GRAL

GRAL je knihovna pro Javu, která je také zdarma a je pro zobrazování grafů a diagramů. V podstatě zkratka GRAL jednoduše znamená vytváření grafů knihovny. Tahle knihovna je určena pro pruhové grafy nebo radarové grafy. [17]

6.1.4. Jgraph

Jedná se o open-source knihovnu, která umožňuje tvorbu a vizualizaci procesních modelů, UML, tvorbu různých typů diagramů. Knihovna je vyvíjena od roku 2000, nejprve byla pouze pro Javu, ale

v poslední době se rozšířila o implementace v JavaScriptu a Flashi. Tuto knihovnu lze využít jak pro komerční, tak i nekomerční aplikace. Hlavním rozdílem komerční licence oproti nekomerční je placená podpora, dále možnost šířit svůj program bez zveřejnění zdrojových kódů a možnost využívat autory dodávané adaptéry pro synchronizaci spojení a dokreslování všech detailů grafů nebo vzorový program, který ukáže, jakým způsobem co nejlépe využít paměť, a zmenšit tak náročnost vyvíjené aplikace na systémové požadavky. Jak původní JGraph, tak současný JGraphX jsou vydávány pod BSD licencí. Základem celé knihovny Jgraph je třída mxGraph, která nám zapouzdřuje všechny podtřídy potřebné k vykreslení grafu. Je to také jednotné rozhraní pro přístup ke všem komponentám grafu. Je děděná z mxEventSource, který v sobě zapouzdřuje správu událostí a jejich spouštění. [18] [1]

Agregované třídy v třídě mxGraph:

- mxGraphModel – modelová třída, která implementuje rozhraní mxIGraphModel, takže má definovány metody na úpravu uzlů jako přidávání, změna, je daný uzel otcem některého uzlu, vrácení kořenového elementu atd. [27]
- mxCell – je třída základního uzlu, může obsahovat sama sebe, obsahuje metody pro zjištění, zda daný bod je hrana nebo uzel, polohu uzlu atd. [27]
- mxGeometry – každá mxCell obsahuje i mxGeometry, která udává jeho grafické ztvárnění, tzn. polohu barvu, relativnost vůči jinému elementu atd. [27]
- mxGraphSelectionModel – je kolekce vybraných uzlů [27]
- mxGraphView – je to taková „cache“ kolekce, kde jsou uloženy všechny uzly s jejich absolutními souřadnicemi a pokud se něco překresluje, tak v rámci optimalizace se snaží aplikace nepřepočítávat celý model, ale vykreslovat ho odsud [27]
- mxStylesheet – touto třídou si lze zde definovat jednotlivé styly, které je pak možné pojmenovat a použít při vykreslování [27]
- mxGraphComponent – je vlastní vizuální třída, která vykresluje graf na nějaké plátno[27]

6.1.5. JFC/Swing

Java Foundation Classes je knihovna Javovských tříd, která zahrnuje i vizuální komponenty které se souhrnně nazývají Swing. Swing je grafický toolkit, který obsahuje jak většinu známých GUI komponent (tlačítka, editační pole, listy, stromy), tak rozvrhovače (Layout managery). Swing dále umožňuje různé tzv. Look & Feel, tzn., že aplikace může vypadat na různých platformách relativně podobně. [12]

6.1.6. JFreeChart

JFreeChart je open-source knihovna pro programovací jazyk Java, který umožnuje vytváření nejruznějších interaktivních i neinteraktivních grafů. Podporuje řadu různých grafů, včetně kombinovaných.

- X-Y grafy
- koláčové grafy

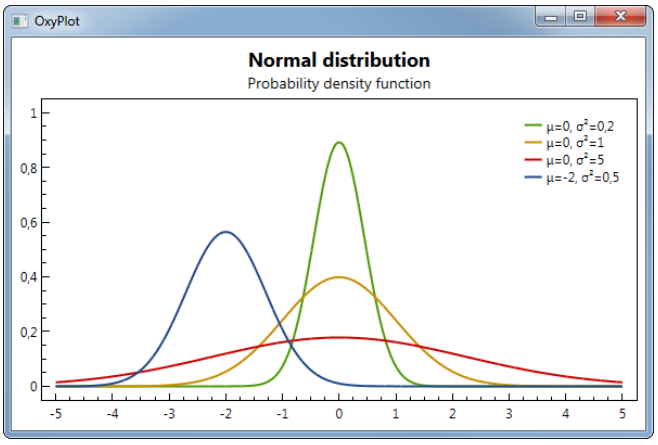
- Ganttův diagram
- Sloupcové grafy
- S jedinou hodnotou (teploměr, kompas, rychloměr)
- Různé specifické grafy (graf vítr, polární graf, bubliny různých velikostí, atd.)

Vykreslování grafů provádí standardním Java 2D API, které zajištují univerzální použití. Pomocí standardních prostředků lze výsledek exportovat do různých formátu, jako jsou png, svg, pdf, jpg, a další. JFreeChart nedokáže zobrazovat dynamické procesy v reálném čase, ale lze v grafu postřehnout změny dat rychlým periodickým překreslováním celého grafu. Tato knihovna tedy neumí zobrazovat 3D grafy. [20] [12]

6.2. Knihovny pro C#

6.2.1. OxyPlot

OxyPlot je cross-platformová knihovna pro vykreslování grafů. Core knihovna je knihovna přenosných tříd, která může být použitá na různých platformách. Ovládací prvky jsou implementovány pro WPF, Windows 8, Windows Phone, Windows Phone Silverlight, Windows Forms, Silverlight, GTK #, XWT, Xamarin.iOS, Xamarin.Android, Xamarin.Forms a Xamarin.Mac. [19]



Obr. 3: Ukázka oxyplotu

6.2.2. D3

D3.js je knihovna, která slouží pro manipulaci s dokumenty založené na datech. D3 nám pomáhá přivést data k životu za použití HTML, SVG a CSS. D3 umožnuje vázat libovolná data do DOM, a pak aplikovat transformace údajů za účelem dokumentu. Například se může D3 použít na generování tabulky HTML z řady čísel. D3 je velmi rychlý, podporuje velké soubory dat a dynamické chování pro interakci a animace. [21]

7. Xamarin

Xamarin umožnuje vytvořit mobilní aplikaci pro tři nejrozšířenější platformy (iOS, Android, Windows Phone). Xamarin je postaven na projektu, který se nazývá Mono. Tento projekt byl původně spravován společnosti Novell, poté fungoval jako samostatný projekt, a nakonec byl zakoupen společnosti Microsoft. Projekt Mono byl založen na ECMA standardech pro C# a CLI, tím pádem je zde možnost psaní mezi platformních aplikací. [22] [26]

7.1. Xamarin.forms

Chceme-li vytvořit mobilní aplikaci, je zde více možností. Záleží, zda chceme aplikaci pouze pro jeden OS nebo pro více najednou. Jestliže chceme vytvořit aplikaci, která není náročná, co se týče množství platformově specifických funkcí, je vhodné použít Xamarin.Forms. Po spuštění aplikace vypadají připravené množiny prvků uživatelského rozhraní jako nativní, tyto prvky nástroj používá napříč systémy. Tento nástroj využívá sdílený kód pro všechny platformy. [22] [26]

8. EKG

Elektrokaridografie je jedna z nejzákladnějších diagnostických metod. Je to diagnostická metoda, která zaznamenává elektrickou aktivitu srdce. Použití je nejvýraznější v kardiologii. Pomoci EKG můžeme zjistit arytmie, ischemické změny v myokardu, slouží také pro kontrolu účinku radiofarmak nebo pro zjištění srdeční choroby. Záznam EKG křivky je elektrokardiograf a výsledná křivka je elektrokardiogram. Pomocí EKG zjišťujeme záznam srdeční činnosti. Srdce je velice důmyslná pumpa, která obsahuje 4 základní části. Pravá a levá síň a pravá a levá komora. Krev vždy čerpáme do síní a ze síní do komor. Tok krve ze síní do komor je usměrňován chlopněmi, ty funguji jako dvoustavové jednosměrné ventily, v ideálním případe proudí krev ze síní do komor vždy jedním směrem. Rozdíl je v srdeční svalovině, v levé části je silnější srdeční svalovina, protože se levá část srdce podílí na větším krevním oběhu. Pravá část se podílí na malém plicním oběhu a prostřednictvím levé části srdce se krev rozbíhá do celého těla. Postupně dochází ke stahu síní a komor. Nejdůležitější na srdci z hlediska EKG je převodní systém srdeční. Je tvořen Hisovým svazkem, z něhož vybíhají dvě Tawarová raménka, která se rozbočují na Purkyňova vlákna, ty ze síní rozvádějí krev do komor. [28]

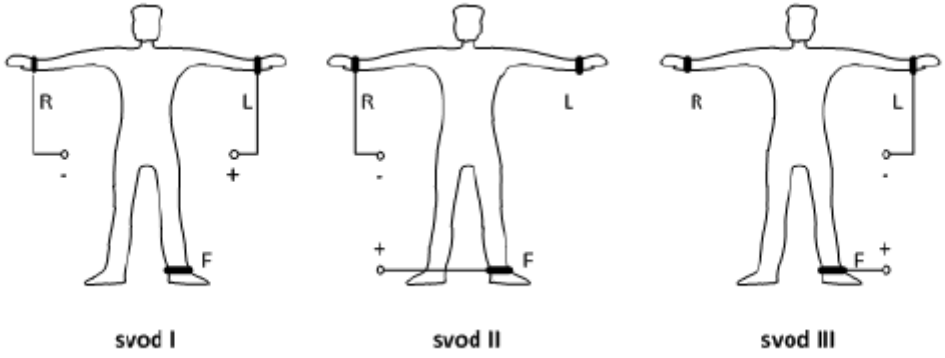
8.1. Měření EKG

Elektrické pole, které vytváří srdce, se šíří vodivým prostředím celého těla, to je důležitý fakt, jelikož jsme schopni snímat EKG z povrchu těla a nemusíme přistupovat k invazivní metodě. U EKG metody se využívá standardizovaného zapojení svodu a celý záznam EKG je standardizovaný. Obecně se využívá trojích svodu. Pro měření EKG se využívá 12-ti svodového EKG a využívají se chlorido-

stříbrné elektrody a to v tom zapojení, že pro končetinové EKG se využívají klipsové elektrody a pro hrudní elektrody balónkové. [28]

8.1.1. Eithovenovy bipolární končetinové svody

Zapojení svodů je vidět na Obr. 4. Značíme svod I, svod II, svod III. Kdy vždy máme dvě elektrody vůči sobě. Svod I je tedy levá paže x pravá paže, svod II máme levou nohu x pravou paže a svod III je levá noha x levá paže.

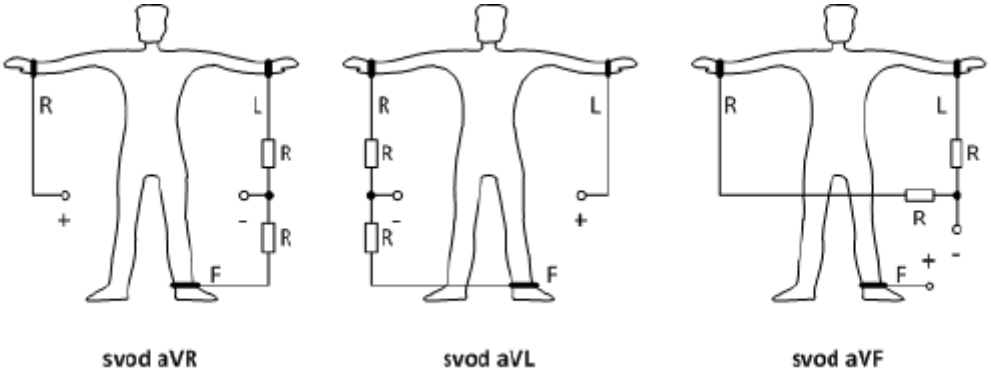


Obr. 4: Eithovenovy svody [28]

8.1.2. Goldbergerovy svody

Stejné elektrody jako u Eithovena. Měřící elektroda se vždy připojuje ke kladné svorce. Ostatní elektrody jsou přes odpor 5kΩ vedeny k záporné-referenční svorce. Zapojení je zobrazeno na

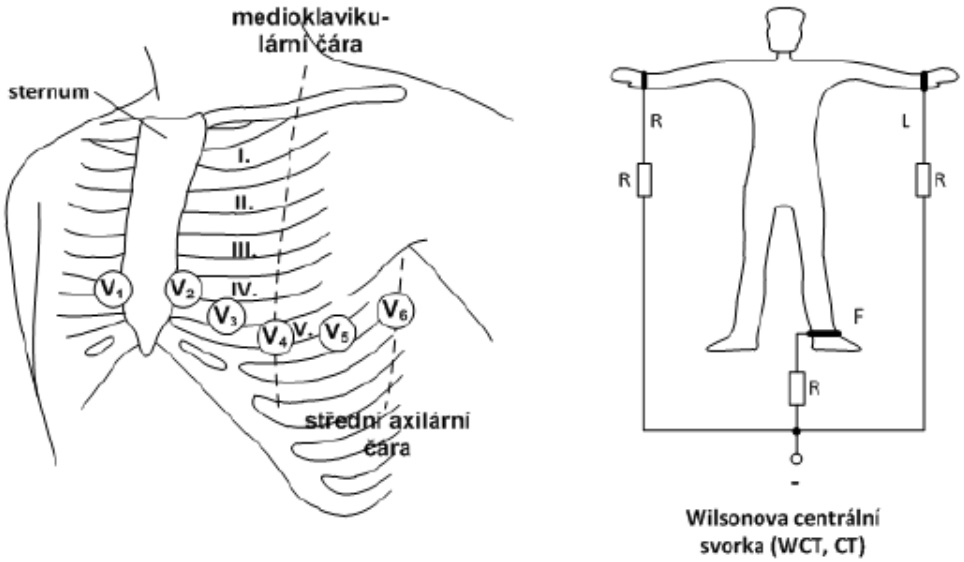
Obr. 5. Označujeme jako aVR, kdy jde o spojení pravé paže a středu levé nohy a levé paže, dále aVL je levá paže se středem levé nohy a pravé paže a aVF je levá noha a střed pravé paže a levé paže.



Obr. 5: Goldbergerovy svody [28]

8.1.3. Wilsonovy svody

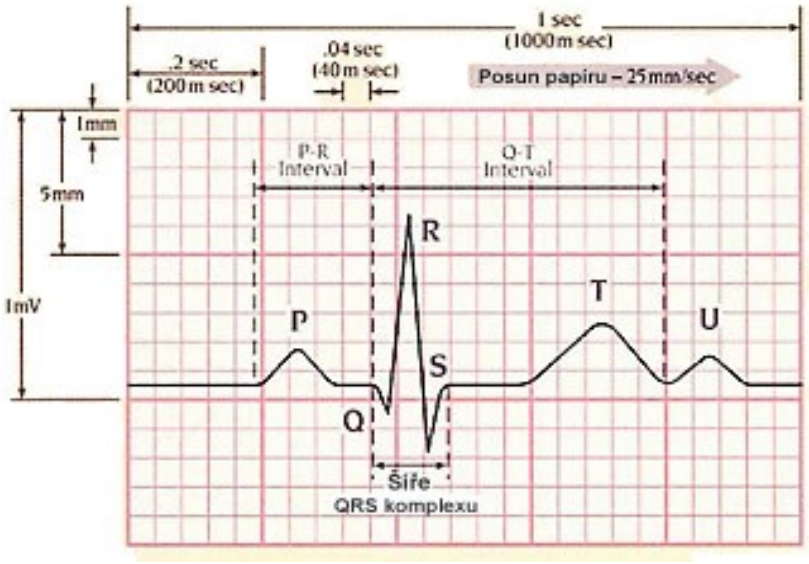
Tyto svody jsou posledním typem svodu 12 ti svodového EKG. Využívají 6 elektrod V1, V2, V3, V4, V5 a V6. Tyto elektrody jsou umístěny na hrudníku a měří se napětí vůči centrální Wilsonově svorce. Rozmístění je uvedeno na Obr. 6. Při rozmísťování se řídíme střední axiální čarou, polohou sterna a medioklavikulární čarou, elektrody v1 až v6 nesou kladný pól a měří se vůči referenční elektrodě, která je daná spojením končetinových elektrod pravé a levé paže a levé nohy jako u Eithovena.



Obr. 6: Wilsonovy svody [28]

8.2. Vznik elektrokaridogramu

Začátek vlny P značí depolarizaci pravé síně a následně i depolarizaci levé síně, což je znázorněno poklesem vlny P na původní hodnotu. Dále následuje depolarizace septa, tam se začíná formovat kmit Q, poté depolarizace hrotů komor a pomalá depolarizace levé komory, toto se objeví jako kmit R, dále pozdní depolarizace obou komor, čímž se doformuje kmit R a nastává kmit S, potom dochází k postupné repolarizaci obou komor.



Obr. 7: Křivka EKG [28]

Základním elementem je izoelektrická linie, což je z matematického hlediska konstantní funkce, rovná čára, která spojuje jednotlivé elementy EKG signálu. Dalším důležitým faktem je vlna, v EKG může být buď pozitivní nebo negativní, která představuje výchylku nad izoelektrickou linii. Z hlediska EKG terminologie rozlišujeme dvouvrcholovou vlnu a vlnu bifazickou. Dvouvrcholová vlna zahrnuje dvě po sobě jdoucí výchylky, které jsou buď obě kladné nebo obě záporné. Bifázická vlna naopak zahrnuje dvě výchylky, s tím že je jedna kladná a druhá záporná nebo naopak. Kmit má podobný průběh jako vlna, jen je více strmější. Kmity jsou v EKG 3 a to Q, R, S. Komplex znamená několik po sobě jdoucích kmitů. Komplex je v elektrokardiografu pouze jeden a to QRS. Úsek, někdy označován jako segment, je doba udávána mezi dvěma elementy na EKG. Existují dva segmenty, segment S-T, který začíná koncem kmitu S a začátkem vlny T, druhým je P-Q, který začíná koncem vlny P a končí začátkem kmitu Q a reprezentuje elektrickou systolu. A posledním elementem jsou intervaly, které jsou dva. P–R interval, který začíná se stahem síní a končí počátkem stahu komor a R – R interval, který udává srdeční frekvenci.

8.3. EKG v praxi

V praxi se pro diagnostické účely využívá 12- ti svodové EKG, které využívá již avizovaných svodů. V praxi se pro toto 12- ti svodové EKG využívá 9 elektrod- 3 končetinové a 6 hrudních. Končetinové elektrody jsou ve skutečnosti 4, kdy ta čtvrtá je na pravé noze, není měřící, je to referenční elektroda, pomocí níž generujeme invertující signál, a která slouží k odstranění stejnosměrné rušivé složky z EKG signálu.

Tak, jak je standardizované měření EKG, tak je i standardizovaný záznam EKG signálu. Jednak citlivost, která je standardně 10 mm /mV, což znamená, že výchylka 1 mV se zaznamená na 10 mm papíru. Další věcí je posun papíru, který je standardně 25 mm za sekundu nebo 50 mm za sekundu. Důležitý amplitudový rozsah signálu, který je pro diagnostické účely se používá pásmo od 0,01-100/150 Hz. Problematičtější je horní rozsah signálu, protože zachází do pásma artefaktů, které jsou

způsobeny v EMG, tím pádem je důležité, aby byl pacient při vyšetření v klidu. Pro monitorovací účely, například pro zátěžové EKG nebo pro dlouhodobě hospitalizované pacienty se horní rozsah snižuje na 30-50 Hz.

9. Popis mobilní aplikace

Tato mobilní aplikace funguje pro vykreslování EKG pacienta z více svodů. Vzhledem k tomu, že je velikost displeje mobilního telefonu omezená, tak se zobrazuje pouze jeden svod, který se zvolí. Aplikaci je možné spustit na všech mobilních platformách. Nejčastější využití je pro zobrazování dat, které jsou naměřené například pomocí vývojového kitu ADS1298. Po výběru pacienta se zvolí svod, který je třeba vykreslit. Dále, pomocí tlačítka START, se začne simulovat online přenos. Je-li třeba záznam zastavit, stačí kliknout na tlačítko STOP. Analýza dat se skládá ze dvou kroků. V prvním se přesně lokalizuje QRS komplex a v dalším se lokalizuje R vlna, která je v EKG signálu nejvíce viditelná. V aplikaci se nachází ještě druhý graf, který slouží pro kontrolu, zda je R vlna správně detekována. Na levé straně obrazovky jsou nastavitelné filtry, kdyby určení R vlny nebylo správné.

Úkolem bylo detekovat z daného signálu QRS komplex, tudíž odfiltrovat šumy způsobené pohybovými artefakty, síťovým rušením, biologickými artefakty. Pro detekování QRS komplexu existuje mnoho metod. Metody založené na diferencích, na číslicové filtraci, na neuronových sítích a na vlnkových transformacích. Tyto metody popisují naprostou většinu algoritmů pro detekci komplexů QRS. Zvolený algoritmus vytvořili pánové Pan and Tompkins (1985).

10. Implementace

Na začátku třídy jsou stanoveny základní proměnné viz Obr. 8. Proměnné jsou typu model pacienta, seznam, číslo, proměnná bool a proměnná algoritmu PanTompkins. Všechny proměnné jsou privátní, tudíž je lze naplnit jen v rámci jedné třídy.

```
public class DisplayECGPageViewModel : BindableBase, INavigationAware
{
    private PatientModel _selectedPatient;
    private List<ChannelModel> _channels;
    private int _tick;
    private bool _isDrawing;
    private PanTompkins algorithm;

    #region PlotModels
    #region ChannelsSelections
    #region BPMLabel
    #region Buttons
    #region Sliders
}
```

Obr. 8: Základní proměnné

Jako další je konstruktor, který je veřejný a název je stejný jako název třídy. Inicializuje proměnnou `PlotModel` vlastní třídou `MyPlotModel`, která obsahuje definice vizuální stránky grafu. Zaregistruje událost pro tlačítko Start-Stop, které obsluhuje vykreslování EKG signálu a nastaví proměnné, které jsou definovány na Obr. 9.

```
  public DisplayECGPageViewModel()
  {
       PlotModel1 = new MyPlotModel();
       PlotModel2 = new MyPlotModel();

       DrawPlotCommand = new DelegateCommand( DrawPlotCommandClick );

       algorithm = new PanTompkins();

       DrawBtnText = "START";
       IsBPMButtonEnabled = false;
       _isDrawing = false;
       _tick = 0;
       CriticalHighPassFrequency = 15.00;
       CriticalLowPassFrequency = 40.00;
       Threshold = 0.35;
}
```

Obr. 9: Konstruktor

Metoda viz Obr. 10 nastaví do proměnné `xAxis` velikost osy x. Když není splněna podmínka, tak to znamená, že proměnná `datamaximum` z `xAxis` je NaN (not a number), tudíž to není číslo. Pokud je podmínka splněná, tak se vstoupí dovnitř, vyčistí se spodní graf a vytvoří se proměnná s jednotlivými body. Do proměnné `length` se vloží počet bodů. `Double x` vytvoří pole, které bude mít velikost proměnné `length`. Jedná se o cyklus, který pole naplní od nuly do `length`, přičemž v každém kroku se naplní jedna část pole.

```
private void CalculateBPM()
{
	var xAxis = PlotModel1.DefaultXAxis;
	if ( !double.IsNaN(xAxis.DataMaximum) )
	{

		PlotModel2.Series.Clear();
		//Preprocess
		var points = ( (LineSeries)PlotModel1.Series.First() ).Points.ToArray();

		int length = points.Length;
		double[] x = new double[length]; // amplitude (Y-axis)
		for ( int i = 0; i < length; i++ )
		{
			x[i] = points[i].Y;
		}
```

Obr. 10: Začátek metody pro výpočet tepové frekvence

Na Obr. 11 jsou proměnné pro algoritmus, je zde zobrazeno, že surová data vstupují do horno-propustního filtru, který slouží k odfiltrování složky velmi nízkých kmitočtů, způsobené mechanickými a elektromagnetickými rušivými vlivy. Výsledek z tohoto filtru vstupuje dále do dolno-propustního filtru, který slouží k odfiltrování složky vyšších kmitočtů a rušivých signálu sítě. Dále vstupuje do diferenciálního filtru, absolutního filtru, který zajišťuje kladnou hodnotu R vlny, integračního filtru, který slouží k nalezení vrcholu R a poté do boolfiltru a nakonec do B2 filtru, tyto dva poslední filtry slouží k prahování a vyhledávání lokálního maxima a minima.

```
double fs = 500;
double hp_fc = CriticalHighPassFrequency;
double lp_fc = CriticalLowPassFrequency;
double[] yhp = algorithm.HighPassFilter( x, length, fs, hp_fc );
double[] ylp = algorithm.LowPassFilter( yhp, length, fs, lp_fc );
double[] ydiff = algorithm.DifferentialFilter( ylp, length );
double[] ysqr = algorithm.AbsoluteFilter( ydiff, length );
double[] yint = algorithm.IntegralFilter( ysqr, length, 60 );
double[] ybool = algorithm.BoolFilter( yint, length, Threshold );
double[] yb2 = algorithm.B2Filter( x, ybool, length, 60 );

List<int> res = new List<int>();
for ( int i = Convert.ToInt32( xAxis.ActualMinimum ); i < Convert.ToInt32(xAxis.DataMaximum ); i++ )
{
	if ( yb2[i] == 1 )
		res.Add( i );
}
```

Obr. 11: Algoritmus

10.1. Filtry

10.1.1. Horno propustný filtr

Přenosová funkce hornopropustného filtru je dána vztahem (1):

$$H(Z) = \frac{\alpha(1-Z^{-1})}{1-\alpha Z^{-1}} \tag{1}$$

Přičemž α upravuje rezonanční frekvenci filtru a vypočítá se pomocí zlomové frekvence fc a vzorkovací frekvence fs podle vztahu (2):

$$\alpha = \frac{1}{2\pi\frac{f_c}{f_s}+1} \tag{2}$$

Z tohoto vzorce je odvozena rovnice pro výpočet hodnoty v diskrétním čase (3):

$$y[n] = \alpha y[n-1] + \alpha(x[n] - x[n-1]) \tag{3}$$

V tomto vztahu je $y[n]$ diskrétní signál, který je na výstupu a na vstupu je signál $x[n]$. Zlomová frekvence horno propustného filtru je $f_c = 15\ Hz$ a vzorkovací frekvence dat je $f_s = 500\ Hz$. Výstup filtru $y[n]$ je vstupem pro následující filtr.

Zdrojový kód pro HP filtr je následující (Obr. 12):

```
public double[] HighPassFilter( double[] x, int N, double fs, double fc )
{
	double[] y = null;
	if ( N <= 0 ) return y;
	y = new double[N];
	double b = 2 * Math.PI * fc / fs;
	double a = 1 / ( b + 1 );
	for ( int n = 1; n < N; n++ )
	{
		y[n] = a * y[n - 1] + a * ( x[n] - x[n - 1] );
	}
	return y;
}
```

Obr. 12: HP filtr

10.1.2. Dolno propustný filtr

Přenosová funkce je daná vztahem (4):

$$H(Z) = \frac{\alpha}{1-(1-\alpha)Z^{-1}} \tag{4}$$

Parametr α je vyjádřený vztahem (5):

$$\alpha = \frac{2\pi\frac{f_c}{f_s}}{2\pi\frac{f_c}{f_s}+1} \tag{5}$$

Odvozená rovnice pro výpočet hodnoty v diskrétním čase (6)

$$y[n] = (1-\alpha)y[n-1] + \alpha x[n] \tag{6}$$

V dolno propustném filtru je zlomová frekvence $f_c = 40\ Hz$. Zdrojový kód je podobný jako u HP viz Obr. 13.

```
public double[] LowPassFilter( double[] x, int N, double fs, double fc )
{
	double[] y = null;
	if ( N <= 0 ) return y;
	y = new double[N];
	double b = 2 * Math.PI * fc / fs;
	double a = b / ( b + 1 );
	for ( int n = 1; n < N; n++ )
	{
		y[n] = a * x[n] + ( 1.0 - a ) * y[n - 1];
	}
	return y;
}
```

Obr. 13: DP filtr

10.1.3. Derivační filtr

V dalším kroku projdou data derivačním filtrem, který je vyjádřený přenosovou funkcí (7):

$$H(Z) = \frac{1}{8}(-Z^{-2} - 2Z^{-1} + 2Z^{1} + Z^{2}) \tag{7}$$

Vyjádření v diskrétním čase (8):

$$y[n] = \frac{1}{8}(-x[n-2] - 2x[n-1] + 2x[n+1] + x[n+2]) \tag{8}$$

Zdrojový kód (Obr. 14):

```
public double[] DifferentialFilter( double[] x, int N )
{
	double[] y = null;
	if ( N <= 0 ) return y;
	y = new double[N];
	for ( int n = 2; n < N - 2; n++ )
		y[n] = ( -x[n - 2] - 2 * x[n - 1] + 2 * x[n + 1] + 2 * x[n + 2] ) / 8.0;
	return y;
}
```

Obr. 14: Derivační filtr

10.1.4. Absolutní filtr

Absolutní filtr funguje tak, že do metody vstupuje pole a vystupuje nové pole, které už nemá žádné záporné hodnoty. Tohle je důležité při negativní polarizaci QRS komplexu.

$$y[n] = |x[n]| \tag{9}$$

Zdrojový kód (Obr. 15):

```
public double[] AbsoluteFilter( double[] x, int N)
{
	double[] y = null;
	if ( N <= 0 ) return y;
	y = new double[N];
	for ( int n = 0; n < N; n++ )
		y[n] = Math.Abs( x[n] );
	return y;
}
```

Obr. 15: Absolutní filtr

10.1.5. Integrační filtr

Integrační filtr slouží k nalezení vrcholu R. Rovnice pro tento filtr (10):

$$y[n+s] = \frac{1}{N}\sum_{m=n-\frac{N}{2}}^{n+\frac{N}{2}} x[m] \tag{10}$$

Ke kompenzaci zpoždění HP a DP filtru slouží proměnná s, která je vypočítaná pomocí $s = \frac{N}{2}$, kde N je počet vzorků, resp. délka integračního okna. Výstup z integrátoru se normalizuje na jednotkový rozsah, pomocí rovnice (11):

$$y[n] = \frac{y[n]}{argmax(y[n])} \tag{11}$$

Zdrojový kód pro tento filtr je na Obr. 16.

```
public double[] IntegralFilter( double[] x, int N, int Nwin )
{
	double[] y = null;
	if ( N <= 0 ) return y;
	y = new double[N];
	int M = Nwin / 2;
	double max = 0.0;
	for ( int n = M; n < N - M - M; n++ )
	{
		double sum = 0.0;
		for ( int i = n - M; i < n + M; i++ )
		{
			sum = x[i];
		}
		y[n + M - 2] = sum;
		max = ( max < sum ) ? sum : max;
	}
	for ( int n = 0; n < N; n++ )
		y[n] /= max;
	return y;
}
```

Obr. 16: Integrační filtr

Na Obr. 17 je zdrojový kód pro výpočet tepové frekvence ze vzdálenosti R vln, který je vypočítán pomocí vzorce (12):

$$BPM = \frac{60000}{vzdalenost*2} \tag{12}$$

Kdy vzdálenost je absolutní hodnota rozdílu poloh detekovaných R vln, vypočítaná pomocí vzorce (13):

$$vzdalenost = |poloha1 - poloha2| \tag{13}$$

Na Obr. 17 je zobrazen zdrojový kód pro výpočet tepové frekence.

```
List<int> res = new List<int>();
for ( int i = Convert.ToInt32( xAxis.ActualMinimum );
          i < Convert.ToInt32( xAxis.DataMaximum );
          i++ )
{
    if ( yb2[i] == 1 )
    res.Add( i );
}
if ( res.Count > 1 )
{
    var dif = Math.Abs( res[0] - res[1] );
    BPMLabel = ( 60000 / (dif*2) ).ToString() + " tep/min";
}
```

Obr. 17: Výpočet tepové frekvence

Pro ověření správnosti detekce R vlny je pod EKG grafem umístěn další graf, který zobrazuje průběh detekce R vln. Zdrojový kód pro tento krok je na Obr. 18.

```
LineSeries resultSeries = new LineSeries();
for ( int i = 0; i < yb2.Length; i++ )
{
	resultSeries.Points.Add( new DataPoint( i, yb2[i] ) );
}
PlotModel2.Series.Add( resultSeries );
ScrollAxis( PlotModel2.DefaultXAxis );
PlotModel2.InvalidatePlot( true );
```

Obr. 18: Ověření detekce

10.2. Uživatelské rozhraní

Uživatelské rozhraní je napsáno pomocí skriptovacího jazyka XAML. K zobrazení uživatelského rozhraní je využita přístupová metoda `data binding`, kdy se propojí hodnota prvku v uživatelském rozhraní s veřejnou proměnnou. Na Obr.19 je pro příklad uveden zdrojový kód pro přenesení hodnoty `slider` do programu.

```
<Slider x:Name="HighPassFilter"
        Grid.Row="3"
        Value="{Binding CriticalHighPassFrequency}"
        Minimum="0"
        Maximum="100"

        />
```

Obr. 19: Slider

Pro vysvětlení je na Obr. 20 zobrazeno provázání uživatelského rozhraní s backend kódem.

```
private double _criticalHighPassFrequency;
public double CriticalHighPassFrequency
{
	get { return _criticalHighPassFrequency; }
	set { SetProperty( ref _criticalHighPassFrequency, value ); }
}
```

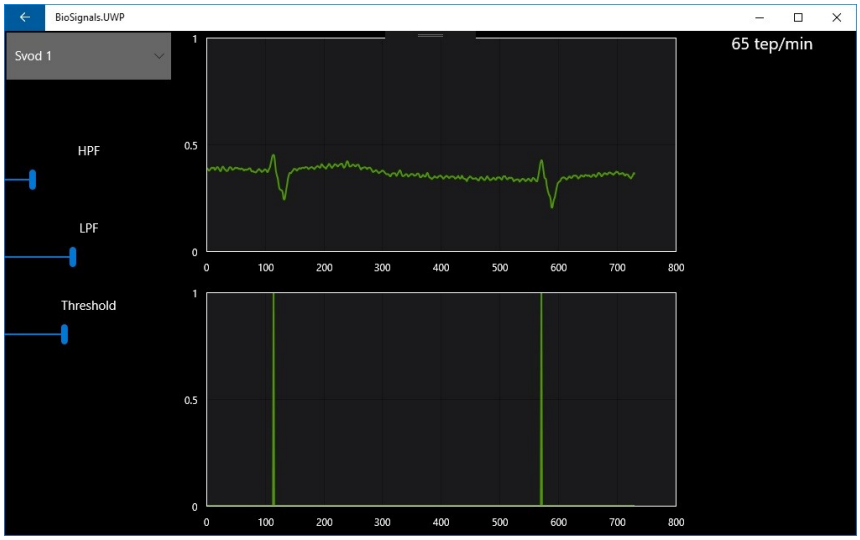
Obr. 20: Deklarace proměnné pro HP filtr

Na Obr. 21 je zobrazeno uživatelské rozhraní první obrazovky s výběrem pacienta. Je zde určena barva textu a pozadí. Je použitý prvek `ListView`, který je navázany pomocí `data binding` se seznamem pacientů.

```
<StackLayout HorizontalOptions="Center">
    <Label Text="Výběr pacienta" TextColor="White" HorizontalTextAlignment="Center"/>
    <ListView HorizontalOptions="Center"
              RowHeight="75"
              SeparatorColor="White"
              ItemsSource="{Binding Patients}"
              x:Name="PacientListView">
        <ListView.ItemTemplate>
            <DataTemplate>
                <TextCell TextColor="White" Text="{Binding Name}" />
            </DataTemplate>
        </ListView.ItemTemplate>
    </ListView>
</StackLayout>
```

Obr. 21: První obrazovka

Druhá obrazovka je rozdělená na tři sloupce. První sloupec slouží jako ovládací panel, ve kterém je možné zvolit svod, který je třeba vykreslit. Jako další je zde tlačítko, které spustí nebo zastaví vykreslování. Poté je možné nastavení tří filtrů, které jsou zmíněny v kapitole 10.1.1, 10.1.2 a parametr prahování bool-filtru. Ve druhém sloupci jsou zobrazeny dva grafy, které jsou vykresleny pomocí Oxyplotu. Nastavení vizuální stránky grafu je definováno ve třídě `MyPlotModel`. Horní graf zobrazuje průběh EKG signálu zvoleného svodu a spodní graf zobrazuje detekci R vlny. Na Obr. 22 je zobrazen snímek z aplikace spuštěné na notebooku Lenovo s operačním systémem Windows 10.



Obr. 22: Druhá obrazovka

10.3. Zpracování dat

Data, která byla použita pro tuto bakalářskou práci jsou v textovém souboru. Data jsou naměřená pomocí vývojového kitu ADS1298. Soubor obsahuje 8 sloupců dat, která byla rozdělena do 8 svodů, což je zobrazeno na Obr.23. Data jsou v jednotkách mV a jsou oddělené tabulátorem.

```
-7,722664E-3    -1,290035E-3    -3,707647E-3    -6,104088E-3    5,195951E-3     -10,105515E-3   -5,040312E-3    1,678181E-3
-7,740259E-3    -1,286650E-3    -3,743839E-3    -6,138849E-3    5,142450E-3     -10,140133E-3   -5,098391E-3    1,605797E-3
-7,759666E-3    -1,274633E-3    -3,791237E-3    -6,175852E-3    5,074787E-3     -10,189581E-3   -5,168438E-3    1,507711E-3
-7,796812E-3    -1,276922E-3    -3,823328E-3    -6,209087E-3    5,052996E-3     -10,238028E-3   -5,189180E-3    1,468945E-3
-7,807112E-3    -1,295424E-3    -3,822374E-3    -6,225109E-3    5,083036E-3     -10,255623E-3   -5,165291E-3    1,493120E-3
```

Obr. 23: Data

Data jsou ve složce `Resources` a mají nastavený příznak `Embedded resources`, což znamená, že se při kompilaci přenášejí společně s aplikací na zařízení. Na Obr. 24 je zobrazen způsob, jak aplikace najde textový soubor s daty na zařízení a načte ho do paměti.

```
var assembly = typeof(Core.Views.DisplayECGPage).GetTypeInfo().Assembly;
Stream stream = assembly.GetManifestResourceStream("BioSignals.Core.Resources." + fileName);
string text = string.Empty;
using (var reader = new System.IO.StreamReader(stream))
{
    text = reader.ReadToEnd();
}
```

Obr. 24: Načtení dat do paměti

Po načtení do proměnné `text` se data rozdělí po řádcích a spočítá se počet svodů, který odpovídá počtu sloupců v řádku. Pak se inicializují modely kanálů a začnou se plnit daty. Data se čtou řádek po řádku a rozdělují se pomocí tabulátorů do proměnné `channelValues`. Poslední vnitřní cyklus prochází rozdělené hodnoty v řádku a přiřazuje je jednotlivým svodům viz Obr 25.

```
var splitByLine = text.Split('\n');
int numberOfChannels = splitByLine[0].Split('\t').Length;

for (int i = 0; i < numberOfChannels; i++)
{
        channels.Add(new ChannelModel("Channel " + i));
}

foreach (var line in splitByLine)
{
        var channelValues = line.Split('\t');

        for (int j = 0; j < channelValues.Length; j++)
        {
                double val = 0;
                double.TryParse(channelValues[j], out val);
                channels[j].Values.Add(val);
        }
}
```

Obr. 25: Zpracování dat

Po naplnění modelů se celý rozsah dat normalizuje do intervalu <0,1>. Na normalizaci slouží funkce `Normalize`, která má 3 parametry. První parametr je pole hodnot, které odpovídá jednomu sloupci, což odpovídá jednomu svodu. Další dva parametry představují dolní a horní hranici intervalu.

10.4. Vykreslení dat

Data jsou v aplikaci vykreslená pomocí Oxyplotu. Velikost plochy pro vykreslování je definována v XAMLu. Přijaté data se zobrazí na grafu, kdy osa x je nastavena podle timeru viz Obr. 26. Data mají vzorkovací frekvenci 500 vzorků za sekundu, tudíž se vykresluje jeden vzorek za 2 ms.

```
private void DrawPlotCommandClick()
        {
            if ( !_isDrawing )
            {
                _isDrawing = true;
                IsBPMButtonEnabled = false;
                Xamarin.Forms.Device.StartTimer( new TimeSpan( 0, 0, 0, 0, 2 ), TimerUpdate );
                DrawBtnText = "STOP";
            }
            else
            {
                _isDrawing = false;
                IsBPMButtonEnabled = true;
                DrawBtnText = "START";
                TimerUpdate(); // To stop timer
            }
        }
```

Obr. 26: Timer

V metodě `TimerUpdate` se volá aktualizace modelu grafu s parametrem vybraného svodu. Po aktualizaci se přepočítává tepová frekvence. Samotná metoda aktualizace spočívá ve vybrání svodu pomocí indexu zvoleného v ovládacím panelu. Poté se na osu vykreslují body, kdy jeden bod se vykreslí co 2 ms a následně jsou spojeny čarou.

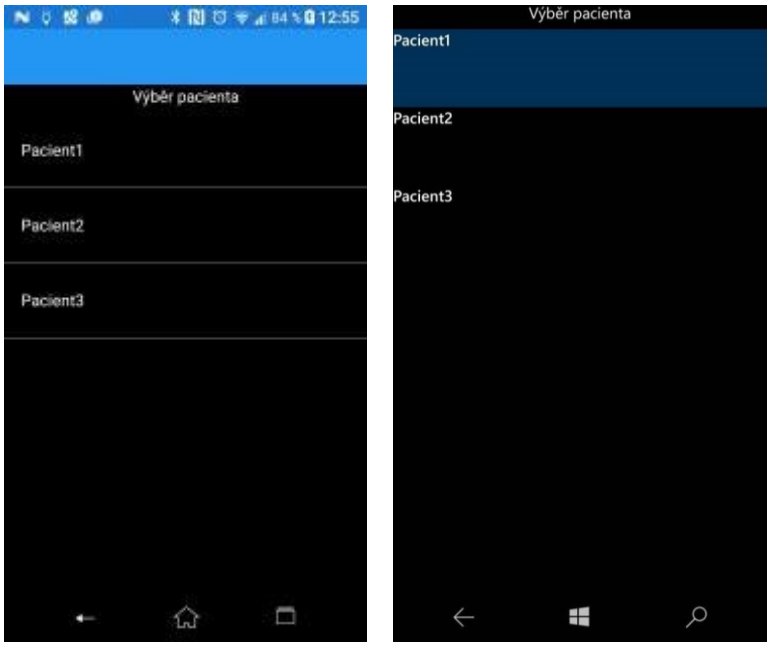
```
private bool TimerUpdate()
{
    if ( _isDrawing == true )
    {
        int selectedChannelIndex = ChannelPicker;
        UpdateModel(PlotModel, selectedChannelIndex);
        CalculateBPM();
        _tick++;
        return true;
    }
    else return false;
}

private void UpdateModel(PlotModel model, int index)
{
    var selectedChannel = _channels[index];
    ScrollAxis( model.DefaultXAxis );
    ( (LineSeries)model.Series.First() ).Points
                                        .Add
                                        (
                                            new DataPoint
                                            (
                                               _tick, selectedChannel.Values[_tick]
                                            )
                                        );
    model.InvalidatePlot( true );
}
```

Obr. 27: metoda TimerUpdate

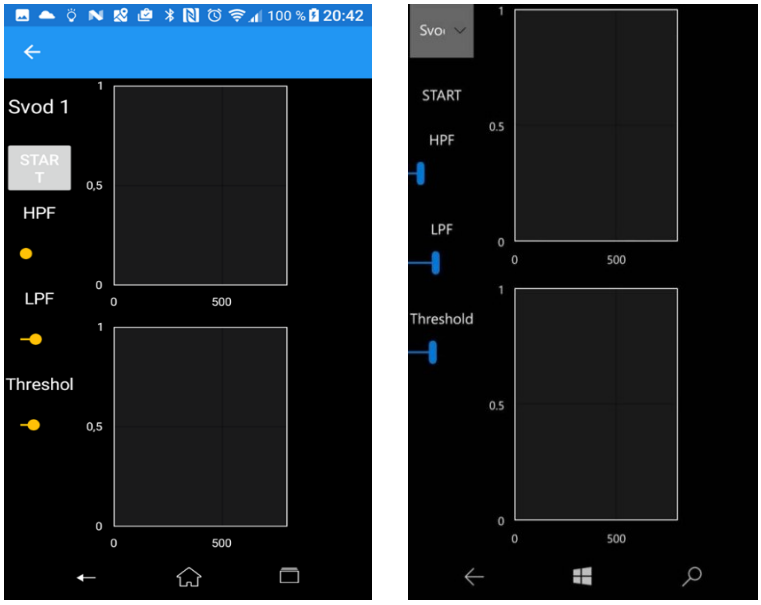
11. Výsledky testování

Aplikace je spuštěna na mobilním zařízení Sony Xperia X Compact s operačním systémem Android. Verze systému android na zmíněném zařízení je 7.0. Druhé zařízení, na kterém je aplikace spuštěna je Microsoft Lumia 650 s operačním systémem Windows 10 Mobile, verze 1607. Na Obr. 28 je zobrazena první obrazovka, kdy je možný výběr pacienta. EKG tří uvedených pacientů, které je naměřeno pomocí vývojového kitu ADS1298, slouží k ověření funkčnosti aplikace. Do aplikace lze nahrát neomezené množství pacientu. Na Obr. 28 vlevo lze vidět aplikaci spuštěnou na Androidu a vpravo na Windows 10.



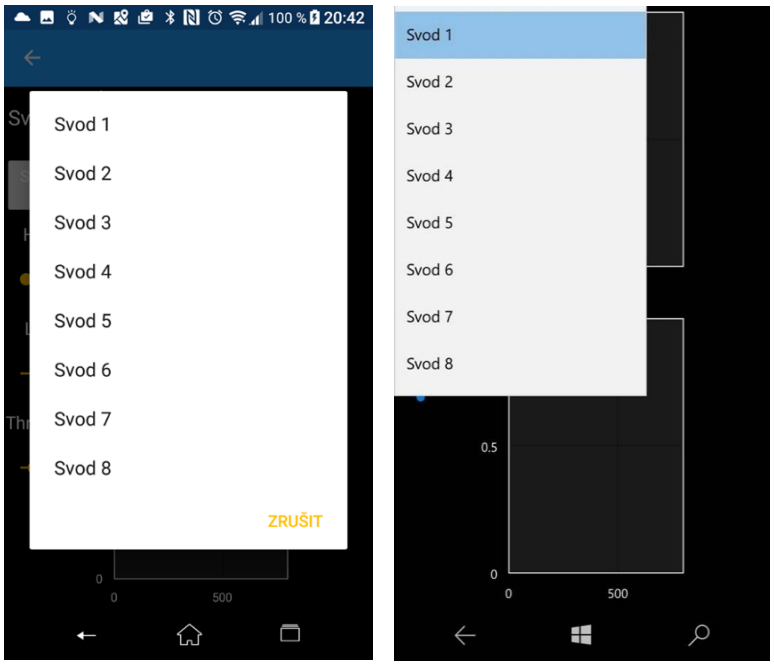
Obr. 28: Výběr pacienta

Na Obr. 29 je druhá obrazovka, pro oba operační systémy, po volbě pacienta před spuštěním EKG záznamu. Je zobrazena možnost nastavení hornopropustného a dolnopropustného filtru. Nastavení je možné od 0 po 100 Hz. Optimální hodnoty jsou u hornopropustného filtru 15 Hz a u dolnopropustného filtru 40 Hz. Poslední možné nastavení je threshold, což je v hodnotách od 0 po 1. Nejpřesnější detekce R vlny je při hodnotě 0,35. Nastavení těchto filtrů je však možné upravovat pro každý svod a každého pacienta zvlášť pro co nejpřesnější detekci.



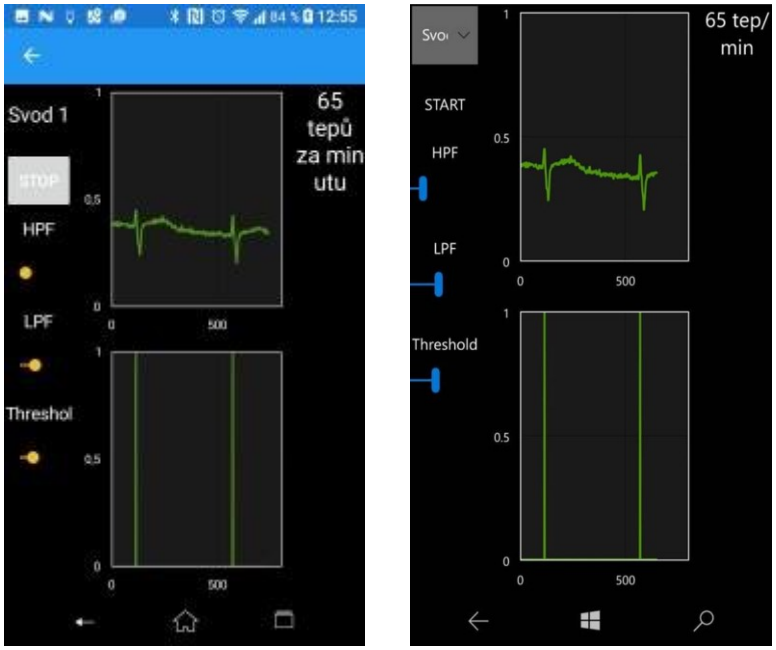
Obr. 29: Druhá obrazovka

Na Obr. 30 je zobrazen výběr svodů pacienta. Opět pro oba dva operační systémy.



Obr. 30: Výběr svodů

Po výběru pacienta a svodu a po stisknutí tlačítka START se spustí EKG záznam a postupně se vykresluje, což je zobrazeno na Obr. 31. Při detekování dvou R vln se vypočítá tepová frekvence a zobrazí se na pravé straně obrazovky. EKG záznam se posouvá a při detekci nové R vlny se tepová frekvence přepočítá.



Obr. 31: Vykreslení EKG

12. Závěr

První čtyři kapitoly teoretické části byly věnovány mobilním aplikacím, mobilním operačním systémům, vývojovým prostředím a programovacím jazykům. Po vysvětlení těchto základních informací, bylo zvoleno vývojové prostředí Visual Studio s programovacím jazykem C#. Aplikace byla vytvořena pomocí Xamarinu pro tři, v dnešní době, nejrozšířenější platformy (iOS, Android i Windows). Další problematikou bylo najít vhodnou knihovnu pro vykreslování grafů. Po zjištění dostupných knihoven je použitý oxyplot. Následující kapitola je věnovaná samotnému EKG, co to je a jak vzniká. Vysvětlení zapojení svodů.

Poslední kapitoly jsou věnovány praktické části. Samotnému popisu mobilní aplikace, jak se ovládá, volitelné nastavení. Poté je v práci prezentován postupný vývoj a zdrojové kódy jednotlivých části aplikace. Je zde uvedený algoritmus pro přesnou detekci QRS komplexu, který se skládá z filtrů, které jsou zapojeny kaskádově. Filtry jsou zde popsány jak matematicky, tak i zdrojovým kódem. Je zde uvedený i postup pro výpočet tepové frekvence, ke kterému je nutná detekce R vlny z EKG křivky.

Aplikace je spuštěna na reálném zařízení Sony Xperia X Compact s operačním systémem Android a na Microsoft Lumia 650 s operačním systémem Windows 10 Mobile, verze 1607. Aplikace umožní výběr pacienta. V tomto případě jsou na výběr 3 pacienti, jejichž EKG je naměřeno pomocí vývojového kitu ADS1298 vedoucím bakalářské práce. Po zvolení pacienta je možný výběr svodu. Pro omezenou velikost displeje na mobilním telefonu je na obrazovku vykreslen pouze jeden svod s možnosti volby, či možnosti přepínání jednoho svodu na druhý. Po detekování minimálně dvou R vln se vypočítá tepová frekvence. Při zobrazení další R vlny, je tepová frekvence automaticky přepočítána. Pomocí tlačítka je možné vykreslování zastavit. Na levé straně obrazovky jsou umístěny nastavitelné filtry pro co nejpřesnější detekci R vlny. Aplikace obsahuje ještě druhý graf, který je pod EKG signálem, který slouží pro kontrolu, zda je R vlna detekována správně.

Výhodou této aplikace je možnost přenosu, či zobrazení EKG signálu mimo dosah počítače nebo pacienta. Další výhodou aplikace je možnost spuštění na všech mobilních platformách. Za nevýhodu je možné považovat, že existuje příliš velké množství zařízení s různým rozlišením a velikostí displeje, a proto je poměrně komplikované přizpůsobit aplikaci pro všechny. Aplikace je v offline režimu, proto by bylo vhodnější, aby se tato aplikace rozšířila na online přenos nebo alespoň o import dat například z internetu. To by vyžadovalo, aby se k datům nepřistupovalo do uložiště, ale byla by použita buď URL adresa nebo FTP server. Pro nahrání více pacientů stačí převést naměřená data do správného formátu, který je zmíněný v 10.3 a vložit do aplikace.

Literatura

[1] *Naučte se Javu – úvod* [online].: 3 [cit. 2015-11-09]. Dostupné z: https://www.interval.cz/clanky/naucte-se-javu-uvod/

[2] KEDZIOR, Roman. *Webové služby pro vizualizaci* [online]. Brno, 2006, 2006 [cit. 2015-11-09]. Dostupné z: https://is.muni.cz/th/60326/fi_m/dipl_prace.txt. MASARYKOVA UNIVERZITA FAKULTA INFORMATIKY.

[3] *LinuxEXPRES - opravdový linuxový magazín* [online]. 2014: CCB, spol. s r. o. [cit. 2015-11-09]. ISSN 1801-3996. Dostupné z: http://www.linuxexpres.cz/

[4] *Linux vs. Unix* [online]. [cit. 2015-11-09]. Dostupné z: http://www.diffen.com/difference/Linux_vs_Unix

[5] KOZEL, Tomáš. *Grafický editor schémat sítě pro informační systém* [online]. Zlín, 2010 [cit. 2015-11-10].

[6] *Informatika a grafika* [online]. [cit. 2015-11-10]. Dostupné z: http://www.gjszlin.cz/

[7] *Programování* [online]. [cit. 2015-11-10]. Dostupné z: http://k-prog.wz.cz/index.php

[8] SKÝBA, Martin. 1997. *Informační technologie pro praxi: sborník přednášek*. Ostrava: Repronis, ^^^svazků. ISBN 978-80-248-3555-6.

[9] SULÍR, Miroslav. *Monitorování provozu operačních systémů a jejich diagnostika*. Vyd. 1. Orlová: Gymnázium a Obchodní akademie Orlová, 2014, 94 s. ISBN 978-80-87477-21-2.

[10] RYCHLÝ, Marek a Jaroslav ZENDULKA. *Modelling of component-based systems with mobile architecture: monograph*. Vyd. 1. Brno: Faculty of Information Technology, Brno University of Technology, 2010, xii, 139 s. ISBN 9788021442115.

[11] GOVE, Darryl. *Programování aplikací pro vícejádrové procesory*. Vyd. 1. Brno: Computer Press, 2011, 416 s. ISBN 9788025134870.

[12] ČIMBORA, Matej. Knihovna jFreeChart pro vytváření grafů [online]. 2012 [cit. 201403-14]. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Marek Grác. Dostupné z: <http://is.muni.cz/th/359468/fi_b/>.

[13] LANG, Jiří. *MULTIPLATFORMNÍ MOBILNÍ APLIKACE* [online]. Brno, 2014 [cit. 2017-01-13]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=119411. Diplomová práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedoucí práce ING. RADEK BURGET, Ph.D.

[14] When to Go Native, Cross-Platform and Hybrid for Mobile. *Program-Ace* [online]. [cit. 2017-01-15]. Dostupné z: https://program-ace.com/press-room/articles/native-cross-platform-hybrid-for-mobile

[15] *JZY3D* [online]. 2009 [cit. 2017-01-15]. Dostupné z: http://www.jzy3d.org/

[16] *Charts4j* [online]. [cit. 2017-01-15]. Dostupné z: https://code.google.com/archive/p/charts4j/

[17] *GRAL Java graphing* [online]. 2016 [cit. 2017-01-15]. Dostupné z: http://trac.erichseifert.de/gral/

[18] *JGraphX (JGraph 6) User Manual* [online]. 2016 [cit. 2017-01-15]. Dostupné z: https://jgraph.github.io/mxgraph/docs/manual_javavis.html

[19] *OxyPlot* [online]. 2015 [cit. 2017-01-15]. Dostupné z: http://www.oxyplot.org/

[20] *To JFreeChart* [online]. 2014 [cit. 2017-01-15]. Dostupné z: http://www.jfree.org/jfreechart/

[21] *Data-Driven Documents* [online]. 2015 [cit. 2017-01-15]. Dostupné z: https://d3js.org/

[22] ŠARLEJ, Filip. *Absolvování individuální odborné praxe*. Ostrava, 2016. Bakalářská práce. VŠB – Technická univerzita Ostrava Fakulta elektrotechniky a informatiky Katedra telekomunikační techniky. Vedoucí práce Ing. Zdeňka Chmelíková, Ph.D.

[23] LACKO, Ľuboslav a Martin HERODEK. [i]ývoj aplikací pro Android. [/i] 1. vyd. Brno: Computer Press, 2015, 472 s. ISBN 978-80-251-4347-6.

[24] VLACH, Karel a Jakub JIRKA. [i]Základy komponentních technologií pro řízení.[/i] Vyd. 1. Ostrava: VŠB - TU Ostrava, 2013. 301 s.

[25] CASTLEDINE, Earle, Jakub MUŽÍK, Myles EFTOS a Max WHEELER. [i]Vytváříme mobilní web a aplikace pro chytré telefony a tablety.[/i] 1. vyd. Brno: Computer Press, 2013, 288 s. ISBN 978-80-251-3763-5.

[26] Xamarin. *Xamarin: Mobile App Development & App Creation Software* [online]. 2017 [cit. 2017-01-16]. Dostupné z: https://www.xamarin.com/

[27] *JavaScript diagramming component,* [online]. England: JGraph, 2000 [cit. 2017-03-12]. Dostupné z: https://www.jgraph.com/index.html

[28] PENHAKER, Marek. *Lékařské terapeutické přístroje*. 1. Ostrava: Vysoká škola báňská Technická univerzita Ostrava, 2007.

Seznam příloh

Příloha na CD/DVD – Zdrojový kód aplikace vytvořený ve Visual Studiu